

**REMARKS/ARGUMENTS**

Claims 1-31 and 43-48 are pending.

Claims 1-5, 7-18, 20-30 and 43-48 are rejected under 35 U.S.C. 102(e) as being anticipated by Jorapur (U.S. 7,299,382). Claims 6,19, and 31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Jorapur in view of Man et al. (U.S. 6,625,760). Applicant submits that all of the claims currently pending in this application are patentably distinguishable over the cited references for the following reasons, and reconsideration and allowance of this application are respectfully requested.

The claimed invention is directed to a method for automatically preventing errors in computer software during its different life cycle phases. The invention initially executes a plurality of software verification tools to verify the computer software. Examples of the software verification tools include, Jtest™, C++Test™, WebKing™, SOAPtest™, CodeWizard™, DataRecon™, SOAPbox™, and WebBox™, from Parasoft Corp. (page 14, lines 6-8). Some examples of different life cycle phases are depicted in FIG. 1, such as, Design phase, Development phase, Deploy phase, and Manage phase. According to the specification, the "system and method of the present invention use error detection as the catalyst for process improvement and error prevention. Each time a specific error is detected, the process is modified to prevent an entire class of errors. To ensure that this methodology becomes a long-lasting part of even the most rushed and chaotic software development processes, AEP includes technologies that automate as many software error prevention practices as possible, including historically difficult and time-consuming practices such as test case creation and verification of custom guidelines/requirements." (Page 5, lines 1-7, underlining added.). The claimed method and system of the present invention therefore "expose different types of mistakes and errors at the earliest possible phase of the software lifecycle." (Page 5, line 12.).

Jorapur does not teach the above invention. Rather, the system of Jorapur generates a plurality of tests 302 (test cases) to test a software. These tests 302 are not the same of the claimed "plurality of software verification tools."

**Appln No. 10/613,166**  
**Amdt date February 19, 2008**  
**Reply to Office action of February 6, 2008**

More particularly, the independent claims 1 and 14 include, among other elements, "executing a plurality of software verification tools to verify the computer software, wherein each of the plurality of software verification tools corresponds to a respective lifecycle phase of the computer software," and "generating verification results for each respective lifecycle phase of the computer software, responsive to executing the plurality of software verification tools and the automatically generated test cases." Jorapur does not teach the above elements.

Regarding the limitation of "executing a plurality of software verification tools to verify the computer software," the examiner cites to col. 4:55-65; col. 6:52-60; and col. 9:45-55 of Jorapur as teaching this limitation. Applicant respectfully disagrees. The above cited passages of Jorapur simply teaches different tests (e.g., test cases) that may be generated in one or more blocks corresponding to one or more parts of the program under test. These tests 302 are generated by a single test generator 301. "The test generator may be a script, an application, or other element configured to generate tests for software. The test generator 301 may return one or more tests 302 corresponding to the application 300. In one embodiment, the test generator 301 produces tests 302 corresponding to modules that are part of the application 300. For example, the test generator 301 may access one or more modules of the application 300 and generate tests 302 including test code configured to interact with at least one of the modules. (Col. 6, lines 41-52, and FIG. 3, emphasis added.). Furthermore, Jorapur is clear about "one or more of the tests used during testing may be generated automatically, either before testing or dynamically during testing. The tests may be from templates. Templates may be stored in a library. One or more tests may be specified for use during testing, for example by a user, by a program, or by default." Col. 2, lines 33-38.).

Neither these individual tests 302, nor the test generator 301 of Jorapur can be constituted as the claimed "plurality of software verification tools," for example, Jtest™, C++Test™, WebKing™, SOAPtest™, CodeWizard™, DataRecon™, SOAPbox™, and WebBox™, from Parasoft Corp. (page 14, lines 6-8). Each of these verification tools is capable of testing different aspects (not portions) of the computer software. For example, CodeWizard™ tool uses Source Code Analysis Technology to parse code (e.g., C and C++ code) looking for specific coding

**Appln No. 10/613,166**  
**Amdt date February 19, 2008**  
**Reply to Office action of February 6, 2008**

patterns that compilers do not detect. (Page 23, lines 18-20). Jtest™ tool automatically tests code construction (white-box testing), tests code functionality (black-box testing), and maintains code integrity (regression testing). (page 10, lines 1-3). SOAPtest™ tool test web services including simple object access protocol (SOAP), etc.

As a result, the generated tests 302 (test cases) of Jorapur can not be constituted as the claimed verification tools and therefore, Jorapur does not disclose "executing a plurality of software verification tools to verify the computer software."

With respect to the limitation of "wherein each of the plurality of software verification tools corresponds to a respective lifecycle phase of the computer software," the Examiner cites to col. 11:30-35, as disclosing this limitation. Applicant respectfully disagrees. The test cases that can be generated by a test generator dynamically may be able to test different functions or portions of the program under test, however, these tests do not correspond to different "lifecycle phases of the computer software," such as, Design phase, Development phase, Deploy phase, and Manage phase. (Id.).

With regard to the limitation of "generating verification results for each respective lifecycle phase of the computer software," as explained above, the tests 302 of Jorapur are not equivalent to the claimed verification tools, and do not correspond to different lifecycle phases and therefore the verification results using those test 302 do not correspond to "each respective lifecycle phase of the computer software."

Consequently, for at least each of the above three reasons, Jorapur does not teach all elements of claims 1 and 14 and therefore, claims 1 and 14 are not anticipated by Jorapur .

Independent claims 27 and 45 include, among other elements, "providing a known error in the computer software, the known error belonging to a class of errors," "providing a plurality of software verification tools each of the plurality of software verification tools corresponding to a respective lifecycle phase of the computer software," "analyzing the known error in the computer software to determine what phase of the lifecycle the error was introduce," and "customizing a verification scope of one or more of the plurality of verification tools that

**Appln No. 10/613,166**  
**Amdt date February 19, 2008**  
**Reply to Office action of February 6, 2008**

correspond to the lifecycle phase that the known error was introduced." Jorapur does not teach the above elements.

First, regarding the element of "providing a known error in the computer software, the known error belonging to a class of errors," there is no teaching of providing a know error in Jorapur. Rather, Jorapur objective is to provide a more automated test generation for making it easier to have a better test coverage. (Col. 2, lines 15-32.). No known error is provided in any of the tests performed by Jorapur's method.

Second, the element of "providing a plurality of software verification tools each of the plurality of software verification tools corresponding to a respective lifecycle phase of the computer software," is not disclosed by Jorapur, as explained above.

Third, the element of "analyzing the known error in the computer software to determine what phase of the lifecycle the error was introduce," is also not disclosed by Jorapur. As discussed above, with respect to claims 1 and 14, the tests 302 of Jorapur are not equivalent to the claimed verification tools, and do not correspond to different lifecycle phases and therefore the verification results using those test 302 do not "determine what phase of the lifecycle the error was introduce."

Fourth, with respect to "customizing a verification scope of one or more of the plurality of verification tools that correspond to the lifecycle phase that the known error was introduced," Jorapur does not teach i) verification scope for each of the verification tools, ii) customization of the verification scope, or finally iii) determining what phase of the lifecycle the error was introduce.


As a result, for at least each of the above four reasons, Jorapur does not teach all elements of claims 27 and 45 and thus, claims 27 and 45 are not anticipated by Jorapur either.

Dependent claims 2-13, 15-26, 28-31, 43-44, and 46-48 are dependent from allowable independent claims 1, 14, 27, and 45, respectively and therefore include all the limitations of their base claims and additional limitations therein. Accordingly, these claims are also allowable over the cited references, as being dependent from an allowable independent claim and for the additional limitations they include therein.

**Appln No. 10/613,166**  
**Amdt date February 19, 2008**  
**Reply to Office action of February 6, 2008**

In view of the foregoing remarks, it is respectfully submitted that this application is now in condition for allowance, and accordingly, reconsideration and allowance are respectfully requested.

Respectfully submitted,  
CHRISTIE, PARKER & HALE, LLP

By   
Raymond R. Tabandeh  
Reg. No. 43,945  
626/795-9900

RRT/clv

CLV PAS779783.1-\* -02/19/08 3:05 PM